REMARKS/ARGUMENTS

Claims 1-15 remain pending in this application and stand rejected. Claims 1-15 are rejected under 35 U.S.C. 112, first paragraph, as containing subject matter which was not described in the application in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. Claim 4 is rejected under 35 U.S.C., second paragraph, as being indefinite for failing to particularly point and distinctly claim the subject which applicant regards as the invention.

Claims 1-2, 4-6, 8-12, and 14-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lauterbach, "Accelerating Architectural Simulation by Parallel Execution of Trace Samples", Tech. Report SMLI TR-93-22, Sun Microsystems Laboratories, Inc., December 1993, pages 1-14, (hereinafter Lauterbach) in view of Applicant's assertions as shown in Fig. 1A. Claims 3 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over the combined teachings of Lauterbach, and Applicant's assertions as applied to claim 1 above, and further in view of Ball, U.S. Patent 5,617,357 issued March 25, 1997. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over the combined teachings of Lauterbach, and Applicant's assertions as applied to claim 1 above, and further in view of Challier, U.S. Patent 6,199,031, and filed August 31, 1998.

Claim 4 is amended to recite in part "The method of claim 1 wherein said state data comprises: program counter contents of set said processor..." Amended claim 4 thus overcomes the rejection made under 35 U.S.C. 112, second paragraph.

In view of the foregoing amendments and following remarks, reconsideration of rejections of claims 1-15 is respectfully requested.

Rejections Under 35 U.S.C. 112

In rejecting claims 1-15 under 35 U.S.C. 112, first paragraph, the Examiner asserts:

> "For example, as shown in Fig. 3A and Fig. 3B and described in the corresponding specification, each low level simulator run one code fragment which may be of determined length or random length. However, without undue experiment, it is unclear what will happen if the destination of a branch instruction is outside of its current code fragment."

Applicants respectfully traverse this rejection. It would be obvious to a person skilled in the art and reading the disclosure of the present application that the code fragments, into which the program is divided, are not dependent on one another, and therefore, the destination branch of an instruction in one code fragment does not fall outside that code fragment. Since the checkpoints, which divide the program into code fragments, include the entire state, and the state data of the processor, such as the register contents of the processor, cache memory contents of the processor, and main memory contents of the processor, as supported, e.g., in pages 6-7 of the original disclosure, all the instructions in the program are present during execution from each checkpoint in the low-level simulator. Hence, branching to a point outside the code fragment that is being simulated does not occur. A number of exemplary embodiments shown in the drawings support this. For example, in describing code fragments 364, 374, and 384, in accordance with one exemplary embodiment, shown in Fig. 3B, the specification provides:

> "This could be especially true in a specific embodiment where checkpoints 360, 370, 380 cover random parts of program 210, where checkpoints 360, 370, 380 divide program 210 into code fragments 364, 374, 384 of random lengths, where code fragments 364, 374, 384 do not overlap, where code fragments 364, 374, 384 are not connected, and where program 210 is executed on each of low level simulators 252, 254, 256 up to a certain point, where each certain point is a point in the program a random length after the corresponding checkpoint 360, 370, 380." (page 9, lines 24-31)

Applicants, therefore, respectfully submit that the original disclosure, as filed, fully meets the requirement under 35 U.S.C. 112, first paragraph. Reconsideration of the rejections of claims 1-15 under 35 U.S.C. 112, first paragraph is thus respectfully requested.

Rejections Under 35 U.S.C. 103

Claims 1-2, 4-6, 8-12, and 14-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lauterbach, in view of Applicant's assertions as shown in Fig. 1A. Regarding claim 1, the Examiner asserts:

> " Lauterbach discloses a method for validating performance and functionality of a processor, comprising the steps of:
> (Claim 1) executing a program on a high level simulator of said processor (the entire execution of the sampled program, page 3, paragraph 3);
> establishing a plurality of checkpoints (the start of the sample instruction trace, page 3, paragraph 3, fifty samples, page 3, paragraph 2);
> saving state data at each of said check points (initial cache state, page 3, paragraph 3)
> running said program on a plurality of simulators of said processor in parallel, starting each of said simulators at a corresponding checkpoint with corresponding state data associated with said corresponding checkpoint (parallel execution, page 10, section 5.0)...."

Applicants respectfully traverse this rejection. Lauterbach fails to disclose or suggest "....establishing a plurality of checkpoints; saving state data at each of said checkpoints; and running said program on a plurality of low level simulators of said processor in parallel, starting each of said low level simulators at a corresponding checkpoint with corresponding state data associated with said corresponding checkpoint", as recited in claim 1, for at least the following reasons.

Lauterbach is concerned with evaluating processor architectural performances not with "validating performance and functionality of processor", as recited in claim 1. As is understood by those skilled in the art, validating the performance and functionality of a processor is an activity that is carried out after the architectural evaluation and design of the processor is, in large part, finalized. For example, Lautrebach discloses:

> "We have developed a viable technique for accelerating architectural simulation to enable number of architectural tradeoffs to be investigated in a short period of time" (page 11, paragraph 4)

> "In order to quickly decide which architectural features are to be included in future processors, we have developed a simulation approach that uses the samples of benchmark program instruction traces..." (abstract)

Applicants thus submit that Lauterbach is not even directed at validating the functionality of a processor. With respect to validating performance, Applicants teach using data in low-level simulators. For example page 7 of the original disclosure provides:

> "In a specific embodiment, each of the low-level simulators is a register transfer level (RTL) model of the processor. In another embodiment, each of the low-level simulators is a VHDL model of the processor. In a different embodiment, each of the low level simulators is a Verilog model of the processor."

The low-level RTL model of the processor provides a very accurate and detailed model of the processor performance. Because Lauterbach is directed at architectural evaluations, Lauterbach does not perform such low level simulations.

Moreover, Applicants teach using such low-level simulators to also validate the functionality of a processor. Applicants submit that Lauterbach is not even concerned and thus fails to teach or suggest anything related to validating the functionality of a processor. For this and the following additional reasons, Lauterbach fails to disclose claim 1.

To reduce simulation times Lauterbach, discloses using "samples of the benchmark program so that only a small portion of the entire instruction trace needs to be simulated". (page 2, paragraph 2). The "start of the sample instruction trace" in Lauterbach is an acknowledgement of the initiation of the tracing of the instructions in the program sample. As best understood, no instructions are executed in Lauterbach's simulations since there is no data associated with the instructions, and the instruction sequence is derived from the trace. For example, on page 5 Lauterbach provides:

> "The SPARC instruction tracer (SHADE(6)) produces a 14-byte structure to describe each instruction traced. For each instruction, the following information is provided:
>
> - program counter (4 bytes)
> - Instruction (4 bytes)
> - Effective memory address (4 bytes)
> - Annulment flag (4 bytes)
> - Branch direction flag (1 byte)"

As is seen from this excerpt, there is no data associated with tracing of instructions. In contrast, in accordance with the present invention, data is used during the execution of instructions in connection with the checkpoints to obtain such information as the state data of the processor. The use of data while executing the instructions in connection with the checkpoints generates a snapshot of the entire state data. This is in contrast with the tracing of instructions of Lauterbach that occurs over time and fails to generate a snapshot of the state of the processor. Checkpoints are described, for example, on page 6, line 29-33, and page 7, lines 1-8, of the original disclosure, and reproduced below for the Examiner's convenience:

> ".....In a specific embodiment, a checkpoint includes a snapshot of the entire state, the state data, of the processor as generated by high level simulator 220

> while executing program 210 during a corresponding interval of the program. In a further embodiment, a checkpoint contains enough state data of the processor such that it is possible to restart a low level simulator 252, 254, or 256, from the same point in program 210, as if the low level simulator 252, 254, or 256 itself had executed all of the instructions before the checkpoint."

The "start of the sample instruction trace", in Lautherbach does not include a snapshot of the entire state, or the state data of the processor as generated by a high level simulator while executing the program. Moreover, because there is no disclosure or suggestion in Lauterbach of low-level simulation, the "start of the sample instruction trace" in Lauterbach also fails to contain enough state data of the processor to enable to restart a low level simulator from the same point in program, as if the low level simulator itself had executed all of the instructions before the checkpoint.

In discussing the limitations of his technique, on page 11 Lauterbach states:

> "Because the method uses the dynamic instruction trace of a program's execution to simulate the architectural simulations, it is difficult to accurately model the behavior of speculative processors..... The instruction traces that we use to generate the trace samples is only able to trace user space instructions. Any activity in system space (system calls, interrupts, traps, etc.) goes unseen, and is therefore not represented....."

Applicants submit that the above limitations of Lauterbach are overcome by the invention of claim 1 because in claim 1, data is used during the execution of instructions to save "data at each of said checkpoints". In other words, because data is used during the execution of instructions in connection with the checkpoints, all processes, whether speculative or not, are accounted for in the present application.

Because Lauterbach fails to disclose "establishing a plurality of checkpoints", it also fails to disclose "saving state data at each of said checkpoints", as recited in claim 1. State data are described, for example, on page 7, line 9-13, of the original disclosure, and reproduced below for the Examiner's convenience:

> "[I]n a specific embodiment, the state data includes (1) register contents of the processor, (2) cache memory contents of the processor, (3) main memory contents of the processor, and (4) branch prediction contents of the processor. In another embodiment, the state data also includes program counter contents of the processor original disclosure...."

The Examiner mistakenly asserts that the initial cache state in Lauterbach discloses "saving state data at each of said checkpoints". However, the "initial cache state" in Lauterbach has no relevance to "saving state data at each of said checkpoints". Lauterbach discloses the following about the cache state:

> "....Rather than increasing the sample size until the misses due to the unknown initial cache state become statistically insignificant, we initialize the cache to the correct state at the start of the sample instruction trace..." (page 3, paragraph 3)

In other words, Lauterbach initializes the cache to correct state at the start of the sample instruction trace to avoid having to increase the sample size to lower the number of cache misses. The "initial cache state" in Lauterbach, however, fails to include, for example, register contents of the processor, main memory contents of the processor, and/or branch prediction contents of the processor. Lauterbach thus fails to teach or suggest "saving state data at each of said checkpoints", as recited, in part, in claim 1.

Lauterbach also fails to disclose "running said program on a plurality of low level simulators of said processor in parallel, starting each of said low level simulators at a corresponding checkpoint with corresponding state data associated with said corresponding checkpoint". Lines 25-33 of page 7 of the original disclosure provide some examples of what a low-level simulator is, as reproduced below:

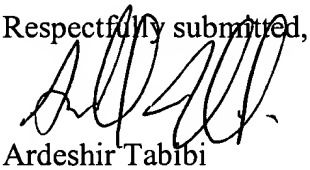
> "In a specific embodiment, each of the low level simulators is a register transfer level (RTL) model of the processor. In another embodiment, each of the low level simulators is a VHDL model of the processor. In a different embodiment, each of the low level simulators is a Verilog model of the processor. In another embodiment, each of the low level simulators is a gate level model of the processor. In another embodiment, each of the low level simulators is a hardware emulator configured with the design of the processor. One type of hardware emulator is a set of field programmable gate arrays (FPGAs), which, when configured with the design of the processor, would map each logical element of the processor into reconfigured hardware of the FPGAs".

Page 10 of Lauterbach, referred to by the Examiner, discloses: "We typically distribute the simulations across fifty SPARC station ECL workstations". There is no disclosure in Lauterbach, however, of using these SPARC station ECL workstations to run low level simulations. Furthermore, there is no disclosure in Lauterbach of "executing a program on a high level simulator of said processor.....and running said program on a plurality of low level simulators of said processor", as recited in claim 1. Moreover, because Lauterbach fails to disclose "establishing a plurality of checkpoints", it also fails to disclose "....starting each of said low level simulators at a corresponding checkpoint with corresponding state data associated with said corresponding checkpoint..."

Claim 1 and its dependent claims 2-10 are thus allowable over Lauterbach either singly, or in combination with Ball and/or Challier. Claims 11-15 are allowable for at least the same reasons as is claim 1.

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested. If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 650-326-2400.

Respectfully submitted,



Ardeshir Tabibi
Reg. No. 48,750

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: (650) 326-2400
Fax: (650) 326-2422
Attachments
AT:deh

60182199 v1